IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Peterson	Docket No.:	ROC920010319US1
Serial No.:	10/068,599	Group Art Unit:	2195
Filed:	02/06/02	Examiner:	Truong, Camquy

TITLE: THREAD DISPATCH MECHANISM AND METHOD FOR MULTIPROCESSOR COMPUTER SYSTEMS

APPEAL BRIEF

Mail Stop APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

This appeal is taken from the Examiner's rejection, set forth in the first Office Action after re-opening prosecution dated 1/25/08. A Notice of Appeal under 37 C.F.R. § 1.191 was filed on 04/25/2008.

REAL PARTY IN INTEREST

International Business Machines Corporation is the Real Party in Interest.

RELATED APPEALS AND INTERFERENCES

This patent application has no related appeals or interferences pending.

STATUS OF CLAIMS

Claims 1-11 were originally filed in this patent application. In response to a first office action dated 05/27/2005, claims 1, 4, and 7 were amended in an amendment dated 09/15/05. In response to a second office action dated 12/12/2005, an RCE and amendment were filed on 05/09/2006. In response to a third office action dated 07/24/2006, and amendment was filed on 10/16/2006 that cancelled claims 8-9 and amended claims 1-7 and 10-11. In the fourth and final office action dated 06/06/2007, claims 1-7 and 10-14 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,912,533 to Hornick in view of U.S. Patent No. 5,293,620 to Barabash *et al.* (hereinafter "Barabash"). The case was re-opened for prosecution and a new office action was sent 1/25/08. In this office action, claims 1-7 and 10-14 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 4,253,144 to Bellamy *et al.* (hereinafter Bellamy). No claim was allowed. Claims 1-7 and 10-14 are currently pending and are at issue in this appeal.

STATUS OF AMENDMENTS

The amendments filed on 09/15/05, 05/09/2006 and 10/16/2006 have been entered. Therefore, the claims at issue in this appeal are claims 1-7 and 10-14 as filed in the amendment dated 10/16/2006.

SUMMARY OF CLAIMED SUBJECT MATTER

Claim 1 recites an apparatus for computer hardware multithreading comprising: a plurality of processors, each processor having hardware support for the capability of executing a plurality of threads (Figures 1 and 4, 110, 112, . . ., 118; p. 14 lines 11-13); a memory coupled to the plurality of processors (Figure 1, 120; p. 8 line 22); and a thread dispatch mechanism residing in the memory and executed by at least one of the plurality of processors (Figure 1, 124; p. 9, lines 13-15), the thread dispatch mechanism determining which of the plurality of processors are idle (Figure 4, 416, 426, . . ., 496; p. 14 lines 13-15), which of the plurality of processors is busy processing a thread but can accept a new thread (Figure 4, 414, 424, . . ., 494; p. 14 lines 11-13), and which of the plurality of processors cannot accept the new thread since it is working on a maximum number of threads the processor can execute (Figure 4, 412, 422, . . ., 492; p. 14 lines 10-11) and, the thread dispatch mechanism dispatching the new thread to an idle processor, if one exists (Figure 5, 522; p. 14, line 18 to p. 15, line 5).

Claim 4 recites a method for dispatching threads in a computer system that includes a plurality of processors that can each support hardware multithreading to execute a plurality of threads (Figures 1 and 4, 110, 112, . . ., 118; p. 14 lines 11-13), the method comprising the steps of: (1) determining the status of each of the plurality of processors, wherein a processor is idle if not executing any threads (Figure 4, 416, 426, . . ., 496; p. 14 lines 13-15), wherein the processor can accept a new thread if busy working on one or more threads but has the capacity to process the new thread (Figure 4, 414, 424, . . ., 494; p. 14 lines 11-13), and wherein the processor cannot accept the new thread if busy working on a maximum number of threads the processor can execute (Figure 4, 412, 422, . . ., 492; p. 14 lines 10-11); and (2) dispatching the new thread to an idle processor, if one exists (Figure 5, 522; p. 14, line 18 to p. 15, line 5).

Claim 7 recites a computer-readable program product comprising: a thread dispatch mechanism (Figure 1, 124; page 9, line 14-15) that determines which of a plurality of processors in a hardware multithreading, multiprocessor computer system are idle (Figure 4, 416, 426, . . ., 496; p. 14 lines 13-15), which of the plurality of processors is busy but can accept a new thread (Figure 4, 414, 424, . . ., 494; p. 14 lines 11-13), and which of the plurality of processors cannot accept the new thread since it is working on a maximum number of threads the processor can execute (Figure 4, 412, 422, . . ., 492; p. 14 lines 10-11), the thread dispatch mechanism dispatching the new thread to an idle processor, if one exists (Figure 5, 522; p. 14, line 18 to p. 15, line 5), wherein each processor can execute a plurality of threads (Figures 1 and 4, 110, 112, . . ., 118; p. 14 lines 11-13); and recordable media bearing the thread dispatch mechanism (Figure 1, 195; page 11, line 16-17).

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

The following single ground of rejection is presented for review on this Appeal:

1. Whether claims 1-7 and 10-14 are unpatentable under 35 U.S.C. §103(a) over Bellamy.

ARGUMENT

Issue 1: Whether claims 1-7 and 10-14 are unpatentable under 35 U.S.C. §103(a) over Bellamy

Appellant traverses the Examiner's characterization of the cited art and the finding of obviousness. The cited art does not teach or suggest the claimed invention. Appellant believes the claims as amended are in condition for allowance and respectfully request the Examiner's rejection be reversed.

Bellamy teaches a plurality of processors that are connected in a hierarchy of levels with provision for communication between the various processors. Bellamy does not even deal with hardware multithreading as claimed herein.

Claims 1, 4, and 7

Claim 1 includes the limitation that the thread dispatch mechanism determines which of the processors is busy processing a thread but can accept a new thread. The combination of the cited art does not teach or suggest to distinguish between a processor that is busy and cannot accept a new thread and one that is busy but can accept a new thread. Looking at a processor to make this distinction concerns hardware multithreading. The cited art does not deal with a processor that has hardware multithreading as recited in the claims.

In the Examiner's latest action, he takes official notice that "thread instructions are commands and that dispatching threads to a processor for execution is well known" (page 3, line 16). While this statement appears to be true, it is misleading and not related to the claim language. Thread instructions may be commands, but **commands are not threads**. The claims refer to the term "thread". While a thread is made of commands, a

thread is not simply a command. Concluding that thread instructions are commands does not mean that it is logical to replace any teaching in the cited art concerning commands with the term thread. These two terms are not interchangeable. A thread is a sequence of related commands or instructions for executing in a sequence. A multi-threaded processor is capable of dealing with multiple threads (plural) at the same time. The claims herein deal with a processor that is working on multiple threads (plural) at the same time. The cited art deals with a single threaded processor. The cited art is only remotely relevant. The Examiner's new office action is no better than the last one that was rejected by the appeal conference and hence the re-opening of prosecution. The Examiner has continued to try to force a round peg in a square hole. The Examiner has not found pertinent art for hardware multi-threading.

For the limitation of "determining which of the plurality of processors is busy processing a thread but can accept a new thread", the Examiner has cited a list of different areas of Bellamy. It is not clear what specific structure in Bellamy the Examiner equates to this claim limitation or for any other limitation for that matter. The Examiner's argument seems to say that a processor that is busy but can accept a new thread is an "engaged processor." The Examiner then states that an engaged processor is "a processor which when it finishes its currently scheduled activities, will then be available for processing" page 3, line 11-12). So by this statement, an engaged processor is a busy processor. The claim says the processor is busy but can accept a new thread. By the Examiner's definition, an engaged processor is a busy processor that can accept a new thread when it is no longer busy. The Examiner's own definition of an engaged processor is one that cannot accept a new thread. A processor that is busy but can accept a new thread is a processor that has processing capability for multiple hardware threads. Bellamy is not concerned with processors that have hardware multithreading and does not teach one of ordinary skill in the art about a busy processor that can accept a new thread.

The Examiner has failed to establish a prima facie case of obviousness under 35 U.S.C. §103(a).

The cited art does not teach or suggest to differentiate between a processor that is idle, one that is busy but can accept a new thread, and one that cannot accept a new thread. In the cited art, there is no hardware support for hardware multithreading, so there are only two states of a processor: idle, meaning the processor is doing nothing; or busy, meaning the processor cannot accept a new thread. In the claims, the dispatch mechanism determines if a processor is busy, whether it is busy but can accept a new thread, or whether it can't accept a new thread. The cited art does not teach or suggest to make this distinction over three possible states of the processor. The cited sections of Bellamy describe systems that are "idle" or "non-engaged". The cited sections do not teach or suggest to differentiate between a processor that is idle, one that is busy but can accept a new thread, and one that cannot accept a new thread. Since Bellamy does not teach or suggest to determine whether a processor is busy but can accept a new thread, claim 1 is allowable over the Bellamy.

Further, claim 1 has the limitation of a processor "having hardware support for the capability of executing a plurality of threads". For this limitation (claim 4 has a corresponding limitation), the Examiner has failed to cite any prior art in the latest office action. The Examiner has merely stated that it is a design choice for the type of processor (page 4, line 1). The claimed invention concerns the function of a multi-threaded processor. The Examiner dismisses the whole field of the invention with a statement of design choice without any citation or argument for each and every claim limitation. The cited art does not teach or suggest that the processors have hardware support to execute a plurality of threads. The Examiner has failed to establish a prima facie case of obviousness under 35 U.S.C. §103(a).

Claim 1 also has the limitation of a processor "cannot accept a new thread since it is working on a maximum number of threads the processor can execute". For this limitation, the Examiner appears to cite the same section of Bellamy concerning an engaged processor. Here again, the Examiner relies on art that does not support his argument. As discussed above, Bellamy does not teach or suggest anything about a processor that can execute multiple threads. The Examiner has failed to establish a prima facie case of obviousness under 35 U.S.C. §103(a). Appellant respectfully requests the board to reverse the Examiner's rejection of claim 1.

Claims 4 and 7 include similar limitations to claim 1 and therefore the same arguments apply to claims 4, and 7. Appellant respectfully requests the board to reverse the Examiner's rejection of claims 1, 4 and 7 under 35 U.S.C. §103(a)

Claims 2, 5 and 10

Claims 2, 5 and 10 depend on claims 1, 4 and 7 respectively, which are allowable for the reasons given above. As a result, claims 2, 5, and 10 are allowable as depending on an allowable independent claim. Further, claims 2, 5 and 10 contain an additional claim limitation that is not taught or suggested by the cited art. For claims 2, 5 and 10, the Examiner cited a list of sections of Bellamy. Again, the Examiner has not shown what structures in Bellamy related to the claim limitations. Appellant has not found anything in the cited section, or in Bellamy in general to support the Examiner's rejection. The cited sections of Bellamy do not teach or suggest "if none of the plurality of processors is idle and if at least one of the plurality of processors can accept a new thread, the thread dispatch mechanism dispatches the new thread to one of the plurality of processors that can accept a new thread." The Examiner states that "choosing a non engaged processor, e.g. a processor which when it finishes it's currently scheduled activities, will then be available for processing of the received command and message" (page 4, lines 9-10). This summation of Bellamy by the Examiner may be correct, but it

does not teach the claim limitation. The claim limitation above deals with a processor that is not idle but can accept a new thread. In Bellamy, as stated by the Examiner, the processor cannot accept the new thread until it is not busy – "when it finishes it's currently schedule activity". Appellant respectfully requests the board to reverse the Examiner's rejection of claims 2, 5 and 10 under 35 U.S.C. §103(a).

Claims 3, 6 and 11

Claims 3, 6 and 11 depend on claims 1, 4 and 7 respectively, which are allowable for the reasons given above. As a result, claims 3, 6 and 11 are allowable as depending on an allowable independent claim. Further, with regards to the rejection of claims 3, 6 and 11, the Examiner states that Bellamy discloses the limitation of "one of the plurality of processors to complete processing a thread" by referring to the same argument about non-engaged processors. The Examiner wrongly interprets the non-engaged processor in Bellamy to be one that is busy but can accept a new thread. This interpretation is simply not supported by Bellamy. Bellamy does not teach or suggest a processor that is busy but can accept a new thread. Appellant respectfully requests reconsideration of the rejection of claims 3, 6 and 11 under 35 U.S.C. §103(a).

Claims 12-14

Claims 12-14 depend on claims 1, 4 and 7 respectively, which are allowable for the reasons given above. As a result, claims 12-14 are allowable as depending on allowable independent claims. Further, with regards to the rejection of claims 12-14, the Examiner states that Bellamy discloses the limitation of "making all the processors busy with a first thread before dispatching an additional thread", citing a list of sections of Bellamy. There is no discussion in Bellamy concerning making all the processors busy with a first thread before dispatching an additional thread to the processor. The Examiner's argument is that Bellamy teaches dispatching tasks first to an idle processor

before an engaged processor. While Bellamy may teach to send tasks to idle processors, Bellamy does not teach or suggest anything about "dispatching an additional thread." The claim is concerned with getting all the processors a first thread before sending an additional thread. While Bellamy does teach about getting a first thread, it does not teach concerning sending an additional thread. In Bellamy, a second task is not sent to the processor until the first is finished. The Examiner's interpretation of the claim and the reading of the prior art effectively removes the claim limitation "dispatching an additional thread" from the claim. Since Bellamy does not teach or suggest to make all the processors busy with a first thread before dispatching an additional thread, Appellant respectfully requests the board to reverse the Examiner's rejection of claims 12-14 under 35 U.S.C. §103(a).

CONCLUSION

Claims 1-7 and 10-14 are addressed in this Appeal. For the numerous reasons articulated above, appellant maintains that the rejections of claims 1-7 and 10-14 under 35 U.S.C. § 103(a) are erroneous. Appellant respectfully submits that this Appeal Brief fully responds to, and successfully contravenes, every ground of rejection and respectfully requests that the final rejection be reversed and that all claims in the subject patent application be found allowable.

Respectfully submitted,

/bretjpetersen/
Bret J. Petersen
Reg. No. 37,417

MARTIN & ASSOCIATES, L.L.C.
P.O. Box 548
Carthage, MO 64836-0548
(417) 358-4700
Fax (417) 358-5757

CLAIMS APPENDIX

1. An apparatus for computer hardware multithreading comprising:

a plurality of processors, each processor having hardware support for the capability of executing a plurality of threads;

a memory coupled to the plurality of processors; and

a thread dispatch mechanism residing in the memory and executed by at least one of the plurality of processors, the thread dispatch mechanism determining which of the plurality of processors are idle, which of the plurality of processors is busy processing a thread but can accept a new thread, and which of the plurality of processors cannot accept the new thread since it is working on a maximum number of threads the processor can execute and, the thread dispatch mechanism dispatching the new thread to an idle processor, if one exists.

2. The apparatus of claim 1 wherein, if none of the plurality of processors is idle and if at least one of the plurality of processors can accept the new thread, the thread dispatch mechanism dispatches the new thread to one of the plurality of processors that can accept the new thread.

3. The apparatus of claim 1 wherein, if all of the plurality of processors cannot accept the new thread, the thread dispatch mechanism waits for one of the plurality of processors to complete processing a thread, thereby becoming a processor that can accept the new thread, and then dispatches the thread to the processor that can accept the new thread.

4. A method for dispatching threads in a computer system that includes a plurality of processors that can each support hardware multithreading to execute a plurality of threads, the method comprising the steps of:

(1) determining the status of each of the plurality of processors, wherein a processor is idle if not executing any threads, wherein the processor can accept a new thread if busy working on one or more threads but has the capacity to process the new thread, and wherein the processor cannot accept the new thread if busy working on a maximum number of threads the processor can execute; and

(2) dispatching the new thread to an idle processor, if one exists.

5. The method of claim 4 further comprising the step of:

if none of the plurality of processors is idle and if at least one of the plurality of processors can accept the new thread, the thread dispatch mechanism dispatches the new thread to one of the plurality of processors that can accept the new thread.

6. The method of claim 4 further comprising the steps of:

if all of the plurality of processors cannot accept the new thread, the thread dispatch mechanism waits for one of the plurality of processors to complete processing a thread, thereby becoming a processor that can accept the new thread, and then dispatches the thread to the processor that can accept the new thread.

7. A computer-readable program product comprising:

(A) a thread dispatch mechanism that determines which of a plurality of processors in a hardware multithreading, multiprocessor computer system are idle, which of the plurality of processors is busy but can accept a new thread, and which of the plurality of processors cannot accept the new thread since it is working on a maximum number of threads the processor can execute, the thread dispatch mechanism dispatching the new thread to an idle processor, if one exists, wherein each processor can execute a plurality of threads; and

(B) recordable media bearing the thread dispatch mechanism.

8. (Cancelled)

9. (Cancelled)

10. The program product of claim 7 wherein, if none of the plurality of processors is idle and if at least one of the plurality of processors can accept the new thread, the thread dispatch mechanism dispatches the new thread to one of the plurality of processors that can accept the new thread.

11. The program product of claim 7 wherein, if all of the plurality of processors cannot accept the new thread, the thread dispatch mechanism waits for one of the plurality of processors to complete processing a thread, thereby becoming a processor that can accept the new thread, and then dispatches the new thread to the processor that can accept the new thread.

12. The apparatus of claim 1 wherein all processors are made busy with a first thread before dispatching a second thread to any processor.

13. The method of claim 4 wherein all processors are made busy with a first thread before dispatching a second thread to any processor.

14. The program product of claim 7 wherein all processors are made busy with a first thread before dispatching a second thread to any processor.

EVIDENCE APPENDIX

An Evidence Appendix is not required for this Appeal Brief.

RELATED PROCEEDINGS APPENDIX

A Related Proceedings Appendix is not required for this Appeal Brief.